TM-0998

Time-homogeneous Parallel Annealing Algorithm (Extended Abstract)

by

K. Kimura & K. Taki

January, 1991

Time-homogeneous Parallel Annealing Algorithm (Extended Abstract)

Kouichi Kimura
kokimura@icot.or.jp

Kazuo Taki
taki@icot.or.jp

Institute for New Generation Computer Technology
1-4-28 Mita, Minato-ku, Tokyo 108, Japan

Abstract

We propose a new parallel simulated annealing algorithm. Each processor maintains one solution and performs the annealing process concurrently at a *constant* temperature that differs from processor to processor, and the solutions obtained by the processors are exchanged occasionally in some probabilistic way. An appropriate cooling schedule is automatically constructed from the set of temperatures that are assigned to the processors. Thus we can avoid the task of carefully reducing the temperature according to the time, which is essential for the performance of the conventional sequential algorithm.

In this paper we propose a scheme of the probabilistic exchange of solutions and justify it from the viewpoint of probability theory. We have applied our algorithm to a graph-partitioning problem. Results of experiments, and comparison with those of the sequential annealing algorithm and the Kernighan-Lin algorithm, are discussed.

1 Introduction

Simulated annealing is a general and powerful technique to solve difficult combinatorial optimization problems [7]. It consists of many iterative *steps*: each modifies the current solution randomly and accepts it with probability $\min\{1, \exp(-\Delta E/T)\}$. Here $-\Delta E$ represents the gain obtained by the proposed modification in terms of the *energy* (objective function) E, and $T > 0$ is the *temperature* which is gradually reduced according to a *cooling schedule*.

Unfortunately, the theoretically optimal cooling schedule, which guarantees the convergence to the optimal solution, proves to be too slow for practical use [4]. Cooling schedules with geometrically decreasing temperatures are often used in applications. To obtain more elaborate cooling schedules is an active area of research [8].

In this paper we propose a new parallel simulated annealing algorithm. It automatically constructs an appropriate cooling schedule from a given set of temperatures.

2 An Annealing Algorithm Parallelized in Temperature

2.1 Outline of the Algorithm

The basic idea is to use parallelism in temperature, to perform annealing processes concurrently at various temperatures instead of sequentially reducing the temperature according to the time.

The outline of the algorithm is as follows. Each processor maintains one solution and performs the annealing process concurrently at a *constant* temperature that differs from processor to processor. After every k annealing steps, every pair of the solutions from the processors with adjacent temperatures are exchanged with some probability p, which is distinct for each pair.

The algorithm can be stopped after any large number of steps and we will find a well-optimized solution on the processor that has the lowest temperature. We refer to $f = 1/k$ as the *frequency of (probabilistic) exchanges* and p as the *probability of exchange*.

Since exchanging the solutions between processors with different temperatures is nothing but changing the temperature for each participant solution, each solution will select its appropriate cooling schedule dynamically through successive competitions with others for lower temperature. However, since the temperature on each processor remains constant, the algorithm itself is *time-homogeneous*. Thus we can avoid the task of carefully reducing the temperature according to the time, which is essential for the performance of the conventional sequential annealing algorithm. In other words, this algorithm automatically decides how many steps should be taken at each temperature: the majority of steps should be devoted to some *critical* temperatures.

However, it is necessary to allocate an appropriate temperature to each processor beforehand. Namely, we have to specify a set of temperatures, from which the algorithm will construct a cooling schedule. This set should be chosen wisely according to the estimation of the equilibrium (static) relation between the temperature and the energy. It must cover the region of temperature, only in which the equilibrium energy varies virtually. Here the concepts of the *scales* by S. White will be useful [10] .

2.2 Probability of Exchange

Investigating the necessary condition which the probabilistic exchange must satisfy, we determine the probability of exchange.

Imagine that the annealing process is performed *independently* at each processor at a distinct constant temperature. Then the distribution of the solution in each processor approaches Boltzmann distribution of the respective temperature [8]. The lower the temperature is, the better the solution that will be found, but after a longer time.

Now we introduce probabilistic exchanges of the solutions between the processors and intend to accelerate the convergence of the solutions so that we can find a better solution at the lowest temperature more quickly.

Let $p(T, E, T', E')$ denote the probability of the exchange between two solutions with energy E and E', at temperatures T and T'. Since we expect a better solution at a lower temperature, we define $p(T, E, T', E') = 1$ if $(T - T')(E - E') < 0$.

On the other hand, if $(T - T')(E - E') \geq 0$, $p(T, E, T', E')$ is *uniquely* determined as follows. In order to accelerate the convergence, a probabilistic exchange of the solutions must not disturb the equilibrium distribution. Hence the detailed balance equation must hold between the distributions before and after the exchange:

$$\frac{1}{Z(T)}\exp(-\frac{E}{T})\cdot\frac{1}{Z(T')}\exp(-\frac{E'}{T'})\cdot p(T,E,T',E') = \frac{1}{Z(T)}\exp(-\frac{E'}{T})\cdot\frac{1}{Z(T')}\exp(-\frac{E}{T'})\cdot 1$$

where $Z(T)$ denotes the partition function. Therefore we obtain

$$p(T,E,T',E') = \begin{cases} 1 & \text{if } \Delta T\cdot\Delta E < 0 \\ \exp(-\frac{\Delta T\cdot\Delta E}{TT'}) & \text{otherwise} \end{cases}$$

$$\text{where} \qquad \Delta T = T - T', \qquad \Delta E = E - E'$$

Note that $p(T, E, T', E') > 0$ for $\forall T\ \forall E\ \forall T'\ \forall E'$. This means that a solution can go through a *non-monotonic* cooling schedule.

This probability is quite different from that of choosing a solution-temperature pair in the systolic statistical cooling algorithm by E. Aarts *et al.* [1]. The advantage of the former is that it does not contain the partition function and hence can be computed easily.

2.3 Monotonic Convergence Property

We verify that each probabilistic exchange of solutions in fact accelerates the convergence of the algorithm.

Let $\boldsymbol{p}$ denote the distribution of the solutions at an arbitrary time. It will change into $\boldsymbol{p}A$ after one annealing step at each processor, or into $\boldsymbol{p}C$ after one probabilistic exchange for each pair of solutions, where A and C are the respective transition probability matrices [6]. Let $\boldsymbol{\pi}$ denote the equilibrium distribution. It can be shown [6] that

$$D(\boldsymbol{\pi}\|\boldsymbol{p}) \geq D(\boldsymbol{\pi}\|\boldsymbol{p}A) \qquad \text{and} \qquad D(\boldsymbol{\pi}\|\boldsymbol{p}) \geq D(\boldsymbol{\pi}\|\boldsymbol{p}C)$$

where $D(\boldsymbol{\pi}\|\boldsymbol{p})$ denotes Kullback-Leibler divergence of $\boldsymbol{\pi}$ and $\boldsymbol{p}$, which represents the discrepancy between them [2]. Here strict inequalities hold unless $\boldsymbol{p} = \boldsymbol{\pi}$. Moreover $D(\boldsymbol{\pi}\|\boldsymbol{p}) \to 0$ follows from the observation in the subsequent subsection.

Hence the distribution of the solutions *monotonically* approaches the equilibrium distribution during the execution.

2.4 Time-homogeneity

The above algorithm is *time-homogeneous*: it has no control parameter to change over time. This has two implications.

Firstly, the behavior of the algorithm is described in terms of a time-homogeneous Markov chain. In general it is an irreducible and acyclic Markov chain over a finite state space. Hence we can easily establish its convergence property [6].

Secondly, in executing the algorithm, we can stop it at any time and examine whether a satisfiable solution has already been obtained. If one has not, we can resume it again for a better solution, and can just continue it as long as we like. In contrast, in the conventional simulated annealing it is necessary to re-schedule the temperature when we resume it, once it has entered the lowest temperature.

3 Experimental Results

We have implemented our algorithm for a graph-partitioning problem on the Multi-PSI/V2 [9], an MIMD parallel machine with 64 processors.

(graph-partitioning problem) Given a graph $G = (\mathcal{V}, \mathcal{E})$, define a *label* on the vertices $\lambda : \mathcal{V} \to \{\pm 1\}$ so as to minimize E, where

$$E = -\sum_{(u,v)\in\mathcal{E}} \lambda(u)\lambda(v) + c \cdot (\sum_{v\in\mathcal{V}} \lambda(v))^2, \quad (c > 0\text{: constant})$$

This is an NP-hard problem [3]. Kernighan-Lin algorithm efficiently gives its approximate solutions [5].

For a random graph G with 400 vertices and 2004 edges, we compare the results given by our algorithm with those by other methods [Fig.1].

(a) Time-homogeneous parallel annealing: All 63 processors performed 20,000 annealing steps each at distinct constant temperatures. The highest and lowest temperatures are determined empirically, and the other temperatures are determined so that adjacent ones have the same ratio. As for frequency of exchanges f, we examined various values ranging from 1/20,000 to 1/2. Each point represents the average over 30 runs with different sequences of random numbers.

(b) Sequential annealing: The cooling schedule consists of exactly the same sequence of 63 temperatures as above. 20,000 annealing steps are performed, which are divided equally between the 63 temperatures.

(c) Simple parallel annealing: Each of 63 processors executes the sequential annealing algorithm described in (b) using a distinct sequence of random numbers. The result is the best solution obtained by them.

(d) Kernighan-Lin: Kernighan-Lin algorithm is repeatedly applied several times until convergence.

We made the following observations from [Fig.1].

1. (a) gives the best solutions for a wide range of the frequency of exchanges: $1/1000 \leq f \leq 1/2$. Hence this algorithm is not sensitive to the value of f except for values that are too small. However a too large value of f incurs a high cost in exchanging the solutions between the processors. The execution time for $f = 1/100$ was less than 8% greater than that for $f = 1/1000$ [6].

2. Since 20,000 annealing steps are relatively small, (b) gives a worse solution than (d). However, in (a), the algorithm probabilistically selects an appropriate cooling schedule with 20,000 steps and gives a better solution.

3. Note that the total number of annealing steps in (a) and that in (c) are the same. (a) outperforms (c) unless f is too small.

4 Conclusion and Future Works

We have proposed the time-homogeneous parallel annealing algorithm, in which an appropriate cooling schedule is automatically and probabilistically constructed from a given set of temperatures.

The behavior of this algorithm is theoretically tractable, since it is described in terms of a *time-homogeneous* Markov chain. In particular we have proved its monotonic convergence property.

We have experimentally observed that this algorithm automatically constructed a better cooling schedule than that which assigned the same number of annealing steps at each temperature. We also observed that this algorithm is robust for the choice of the frequency of exchanges.

The following require further investigation.

(i) How many processors should we use?

(ii) How should we assign temperatures to the processors?

(iii) How do we find the *optimal* frequency of exchanges?

(iv) Does this algorithm probabilistically select the *theoretically best* cooling schedule, the *best* assignment of the annealing steps to each temperature?

5 Acknowledgments

We would like to thank N. Ichiyoshi, K. Rokusawa, and E. Sugino for valuable discussions.

References

[1] E. H. L. Aarts *et al.*, "Parallel Implementations of the Statistical Cooling Algorithm," *Integration*, 4, (1984).

[2] S. Amari, "Differential Geometric Methods in Statistics," Lecture Note in Statistics **28**, Springer-Verlag, (1985).

[3] M. Garey and D. Johnson, "Computers and Intractability, A Guide to the Theory of NP-Completeness," Freeman, New York, (1979).

[4] B. Hajek, "Cooling Schedule for Optimal Simulated Annealing," *Math. Oper. Res.*, 13 (1988).

[5] B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell. sys. tech. J.*, 49, (1969).

[6] K. Kimura and K. Taki, "On a Time-homogeneous Parallel Annealing Algorithm," *ICOT Technical Report*, 565 (*in Japanese*) (1990).

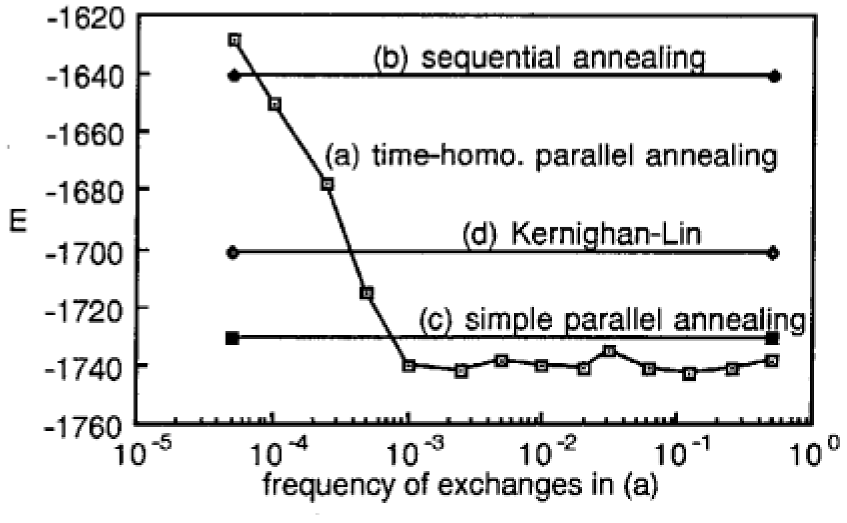
[7] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecci, "Optimization by Simulated Annealing," *Science*, vol.220, no.4598 (1983).

[8] P.J.M. van Laarhoven and E.H.L. Aarts, "Simulated Annealing: Theory and Applications", Reidel, (1987).

[9] K. Nakajima *et al.*, "Distributed Implementation of KL1 on the Multi-PSI/V2", *Proc. 6th Int. Conf. on Logic Programming* (1989).

[10] S. R. White, "Concepts of Scales in Simulated Annealing," *Proc. IEEE ICCD* (1984).

Fig.1. Energy vs frequency of exchanges


E
(b) sequential annealing
(a) time-homo. parallel annealing
(d) Kernighan-Lin
(c) simple parallel annealing
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
frequency of exchanges in (a)